Autonomous Object Moving Vehicle

By

Chenliang Li

Honglu He

Siping Meng

Final Report for ECE 445, Senior Design, Fall 2018

TA: Amr Martini

11 December 2018

Project No. 32

Abstract

We designed a small automated robot which could grab and shift objects after receiving the command given by the user. Our robot has a robot arm to pick objects, and the lower part is a chassis vehicle for moving. We installed a camera and a LiDAR sensor in the front of the robot to help it detect objects and obstacles. We also implemented various algorithms in the microcontroller to give this robot capabilities to find the moving path automatically and to place the object in the right place under an acceptable tolerance. Besides, our robot is designed to accomplish the task offline and automatically so that user does not need to wait for the robot to finish its job. Our team worked together with Team 19, and we were in charge of the lower part of the vehicle.

Contents

1 Introduction

For a long time, the idea of robots doing chores around the house has long captured people's imaginations [1]. However, a generalized servant robot is still a rare sight in the home today. Besides, although the smart house can support remote control to turn the air conditioner on or off through host's mobile phone, there are still numerous detailed situations that are hard to accomplish by merely using programmed command and smart appliances nowadays in the market. We still need a robot to help us arrange indoor objects and shift them for various needs. Until recently, a smart domestic robot dog, SpotMini, created by Boston Dynamics, shows up and demonstrates its powerful abilities to open the door and shift a cup of water just like our project goal. The task's complexity which SpotMini can do is nearly reach the hardness level of daily human tasks. However, there is some deficiency for SpotMini. The size of SpotMini is too big to recognize relatively small obstacles underneath. It still has a high possibility to collide with obstacles (like wall or furniture) when it turns around and even fails in a real indoor environment. Therefore, we are looking forward to a solution for those disadvantages. To be more specific, to redesign a smaller robot which can provide similar functionality with more accurate position precision. Our team worked with team 19 to accomplish this challenging concept. Our team focused on the wheel motor control, localization and mapping, and path planning. This paper will explain how we achieved our goal by calculation and experiment.

2 Design

2.1 Block Diagram

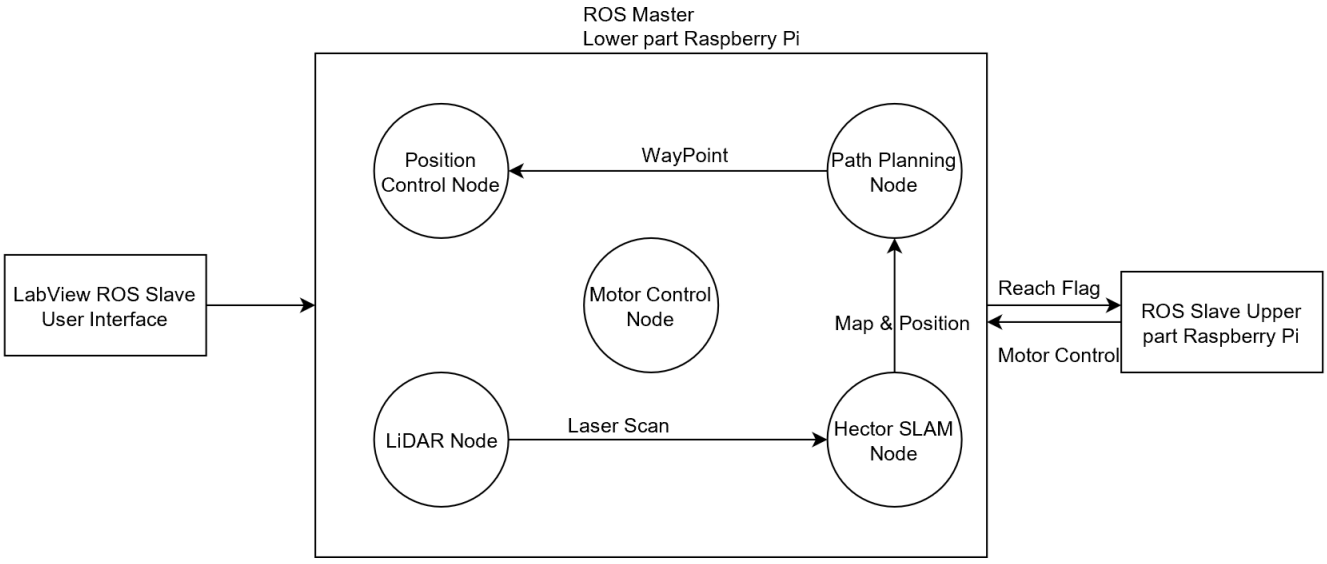

Figure 1 Block Diagram

Our overall design used Robot Operating System (ROS) as a tool to connect different nodes on same machine or different machines. The user interface was a LabView program which can send the destinations to the robot, including the original object position and the position we want our object to be

placed. We used 2 Raspberry Pis as our microprocessors, one for upper part and one for lower part. The lower part Raspberry Pi is the ROS master which contains all nodes related to lower part. The upper part Raspberry Pi communicates with lower part Raspberry Pi to check if the destination is reached or not, as well as take over the control of both wheel motors.

Inside the ROS master, the LiDAR node is used to publish Laser Scan message (angle and distance of each laser beam). Hector SLAM node is to subscribe to the Laser Scan message and output position of the robot and map of the environment. Path Planning node utilizes both information from Hector SLAM to produce an optimal path to the user-specified destination from LabView. Position Control Node takes each waypoint in the path and controls the vehicle to go to the waypoint directly. Motor Control Node is used to subscribe message from upper part Raspberry Pi so that the robot can adjust its position and angle.

2.2 Physical Design

The physical design of this robot is chosen to allow flexible moving with a relatively small size and to place all sensors in appropriate positions to work. Our robot consists of a Three-layer chassis vehicle with two drive wheels and one omnidirectional wheel. The dimensions of the chassis vehicle are 250mm x 205mm x 103mm. We choose only two wheels rather than four in attempts to reduce the cost and to allow the robot to do pivot turns more easily.

We placed a robotic arm on the top of the chassis (on the Third layer) actuated by two motors to grip items. On the top layer, we will place a LiDAR in the front of the vehicle to detect obstacles when the robot is moving. However, the arm and camera will be put on the back, because we do not want other physical components to block the vision of LiDAR otherwise the obstacle detection will be undermined. When the robot arrives its destination and is going to pick up an item, it needs to turn around first so that its robotic arm and camera will face to the item. We will put the remaining hardware components on the first and second layers of chassis, including the microcontroller, PCBs.

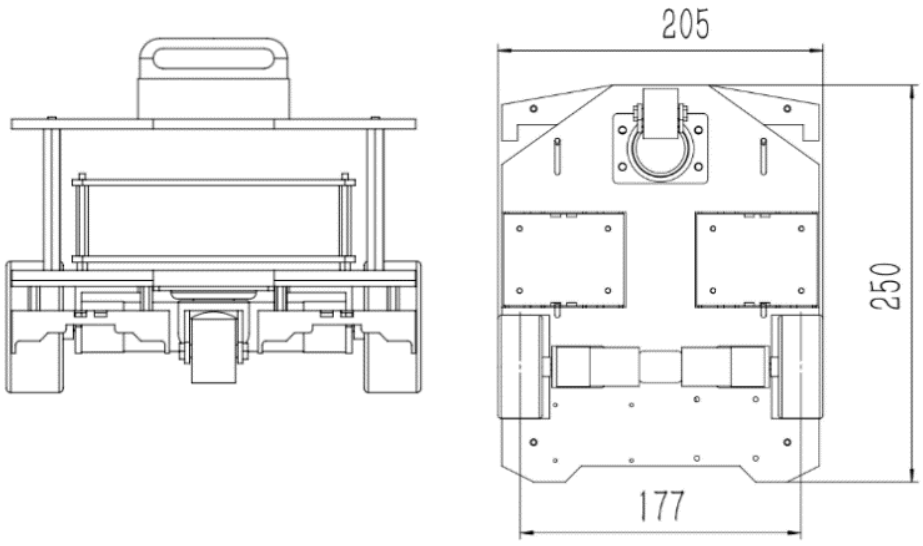

Figure 2 Physical overview of chassis vehicle (unit: mm)

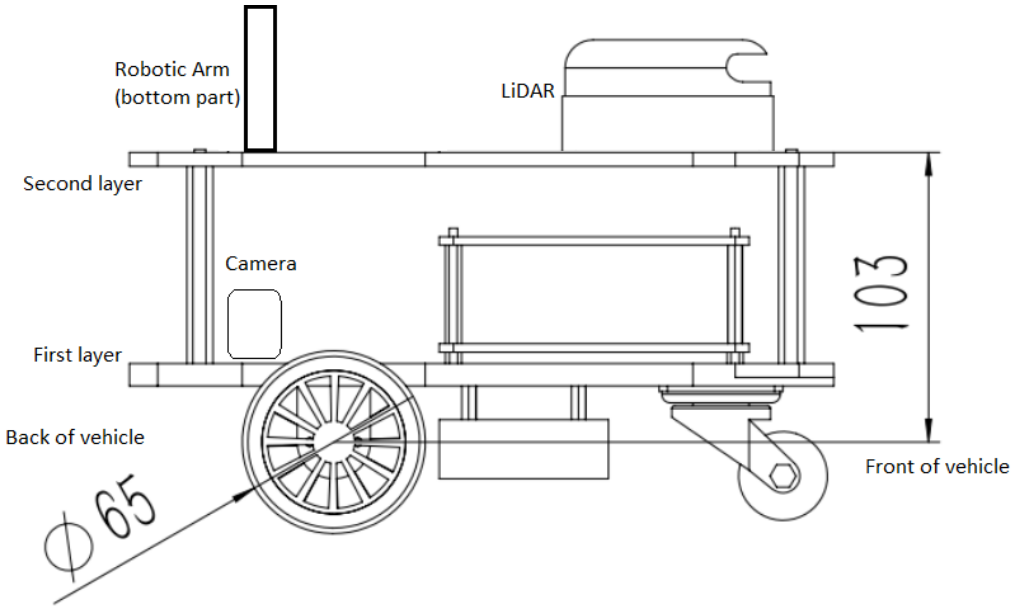

Figure 3 Relative positions of arm, LiDAR, and camera (unit: mm)

2.3 Modular Design

2.3.1 Motors, PCB and Motor Control

We chose brushed motor rather than the brushless motor because it is inexpensive, steady, and easy to control: we can use voltage to control the speed of the motor, and we can also change the voltage direction to change the rotation direction of the motor. We designed a PCB for our controlling motor system, centered on TB6612 motor controlling chip. Taking into consideration of heat dissipation and current intensity, we meticulously designed the wires on the board. The TB6612 chip provides two H-bridge circuits controlled by four logic voltage signals and two PWM controller for the output voltage amplitude for the motor. We also designed protection circuit for sudden current change due to the motor connected to the PCB.

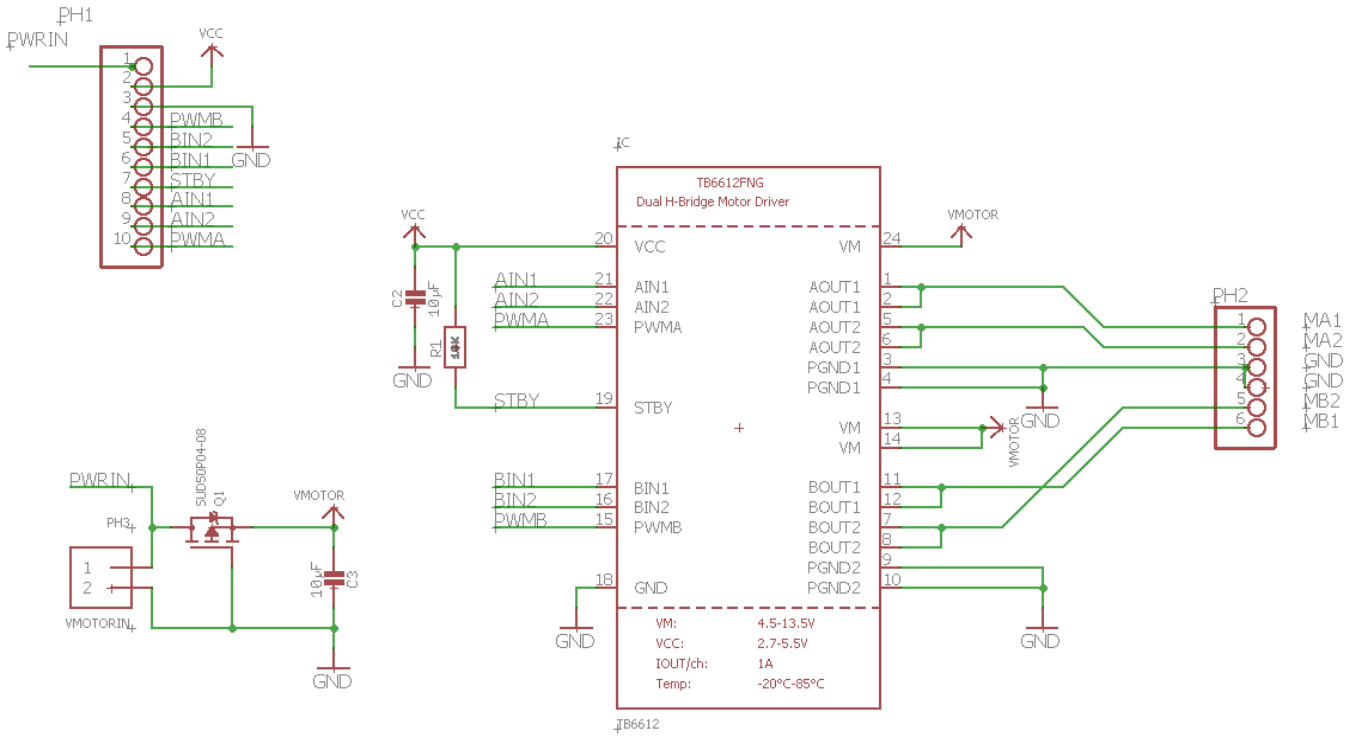

Figure 4 PCB Schematic

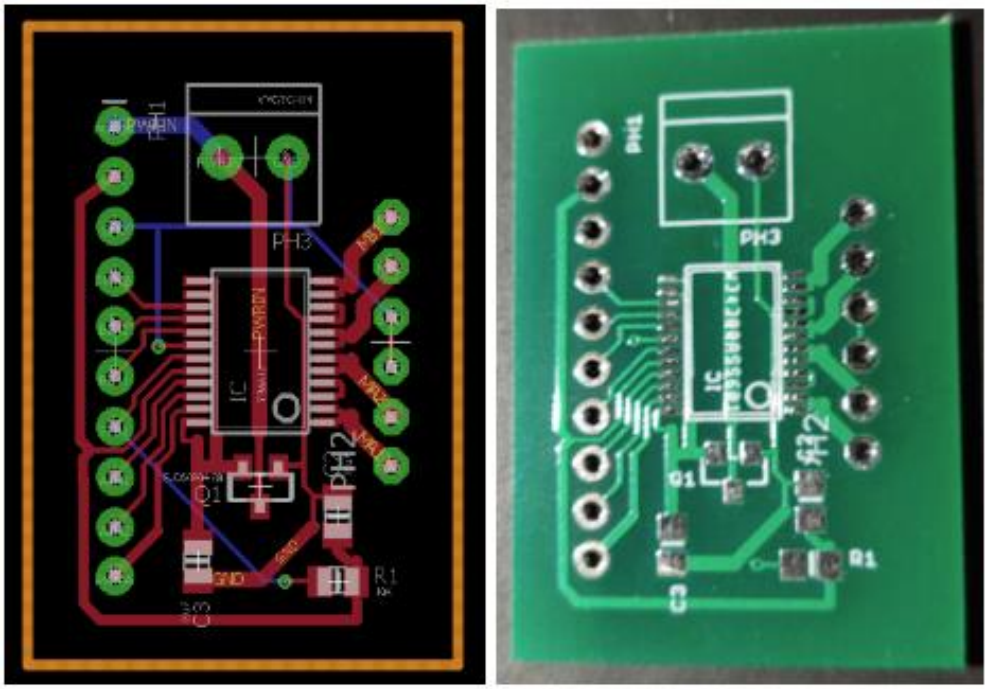

Figure 5 PCB Board Layout

2.3.2 Localization

Starting from the home position (0,0), Hector SLAM publishes Pose message, which contains the world coordinate (x,y,z) in meter and angle in quaternion form (x,y,z,w) [2]. Regarding world coordinate, z is always 0 in our case. Moreover, the position from Hector SLAM is always the center of the LiDAR. Since Euler Angle contains Roll, Pitch, and Yaw, but all we need is Yaw angle due to the flat surface. According to the conversion equation from a quaternion to Yaw Angle, we used the following equation [3]:

$$\psi = \mathrm{ATAN2}\big(2(\mathrm{wz} + \mathrm{xy}), 1 - 2(\mathrm{y}^2 + \mathrm{z}^2)\big)$$

Equation 1: Quaternion Transformation

where ψ is Yaw Angle in rad, ATAN2 is 2-argument arctangent.

2.3.3 Map

After Hector SLAM algorithm successfully sends out occupancy array, the map module will start to convert this one-dimension array to two-dimension array for future calculation and visualization. The occupancy array contains three values: -1 means not sure, 0 means space and 100 means obstacle. Here we used python numpy reshape for conversion, and then slice out the appropriate region for our project. Doing slicing is because the entire map contains 2048*2048 points, which is a quite large map, but we only need a smaller map that is in front of the robot's initial location. So we decide to slice out a 200 * 100 map with robot's initial location at (100,0). Then the map module will loop through all the points and save obstacle coordinate in a set.

While the module adding detected object coordinate into a set, this module will automatically populate all the obstacles according to a given robot size variable.

```
BUILDMAP(map, robotsize):
    map <- convert_to_2_D(map)[1024:1124,974:1074] #slicing out a 200 * 100 map
    for each point in the map:
        if point is wall:
            for all the n_point around point within robotsize:
                if n_point in bound: add n_point into set
```

Figure 6 Pseudocode for buildmap()

2.3.4 Path Planning

Path planning is the key in our project that can make the robot move around smoothly. The goal is to ensure this robot could walk along the shortest path while staying collision-free. After map module builds up a map, path planning module will accept a start and end point, using that map, and give back an optimum path. We choose A* algorithm with Euclidian heuristic since our map is equal cost and it is easy to realize. We have considered other methods such as probability roadmap, but for convenience, we stick on A* in this project. In this module, because our map is grid-based, the path can only fit into the grids. Using a grid-based map will cause zigzag movements, but our module fixes this inefficiency by compress the path before returning it to the control module.

We can accomplish this with a mathematical approach. From the starting point, our module will go through all the following points and check if it can be compressed by deleting all nodes between them. Intuitively, our module draws a line between two points, and for each point that has a distance less or equal to 1 to the line, the module will keep this point if there is any obstacle.

This is the distance from point B to line AC:

$$Distance = \frac{norm\big(cross(C - A, A - B)\big)}{norm(C - A)}$$

Equation 2: Point to Line Distance

```
COMPRESS(path):
    if path length <= 3: return
    base, cur, next = path[0], path[1], path[2]
    for i from 2 to path length:
        for each distance(point, line(base,next)) <= 1:
            if wall exists: signal = 1
        if signal:
            base, cur, next = cur, next, nextpoint in path
        else:
            pop cur from path
            cur, next  =  next, nextpoint in path
    return path + end point
```

Figure 7 Pseudocode for path compression

2.3.5 Position Control

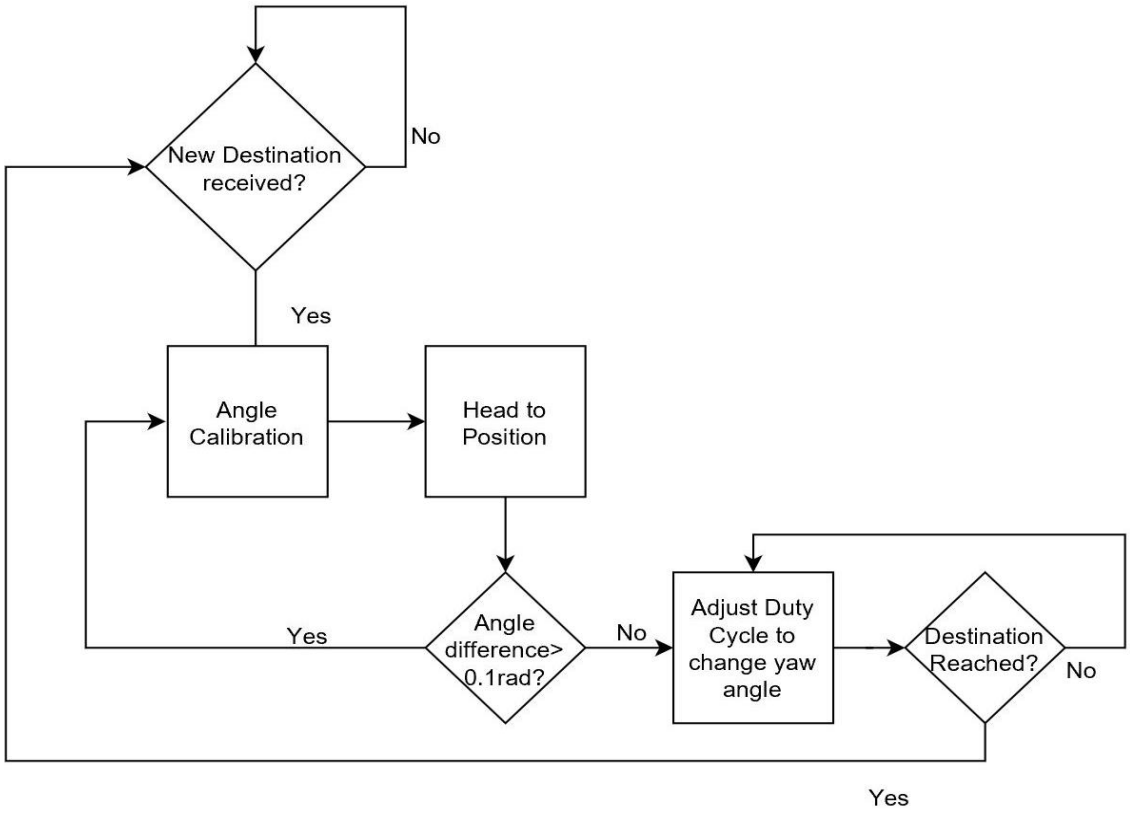

Figure 8 Position Control Diagram

Position control is used to receiving waypoints in the path planned by path planning node. This node will keep polling until a new waypoint is received. The first step is to calibrate its yaw angle to face the destination; after that, it will start going straight and regularly check its distance to the destination and its angle as well. If the angle difference between a current angle and desired angle (facing the destination), it will stop and calibrate the yaw angle again. If the distance to the destination is less than 10cm, it will stop and publish a message to let the path planning node know that it has arrived at this waypoint.

2.3.6 User Interface

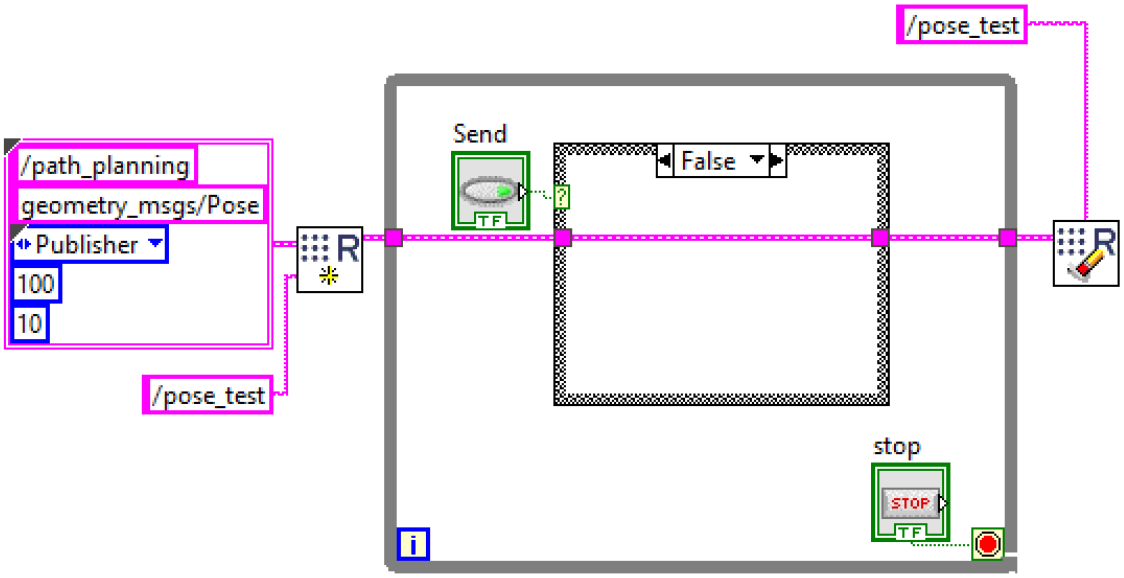

Figure 9 User Interface LabView Block Diagram

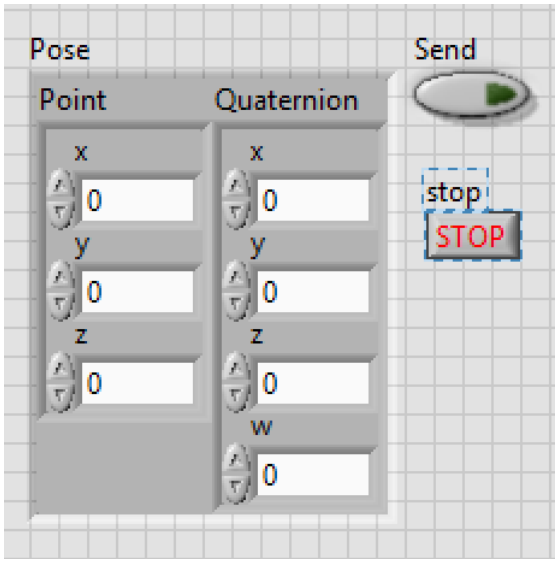

Figure 10 User Interface LabView Block Diagram

On the Windows laptop, a LabView program for the user is used to send destination information to the robot. This program is also a ROS node, and publishes Pose message reading from the user, and the user only need to input x and y information.

3 Design Verification

3.1 Motor Control Test

For the motor test: We connected the motor to the power supply in the lab, input different direction voltages to test two side rotation. Then we tested the speed of the robot under different voltage with different loads. The robot needs to move at 4.2cm/s with maximum 6.7 kg loads.

For the PCB test: We input mock logic voltage signal and mock PWM signal from the generator in the lab. By check the output voltage of the PCB, we could verify if logic voltage can change the motor voltage direction or if the PWM duty cycle can change the motor voltage by the multimeter.

A combinational control test with Raspberry Pi: By connecting GPIO pins on the Raspberry Pi and pins of PCB and change the output of PWM's duty cycle and logic voltage by a program in python, then verified the output voltage amplitude and direction by the multimeter.

3.2 Localization Test

We put our LiDAR on a dolly and let the dolly go through a predefined loop on a rail, and then we analyzed the map and trace that the LiDAR generated to see if the LiDAR's accuracy. As the result shows, our localization has very high accuracy with an error less than 0.04m, and the loop is closed.

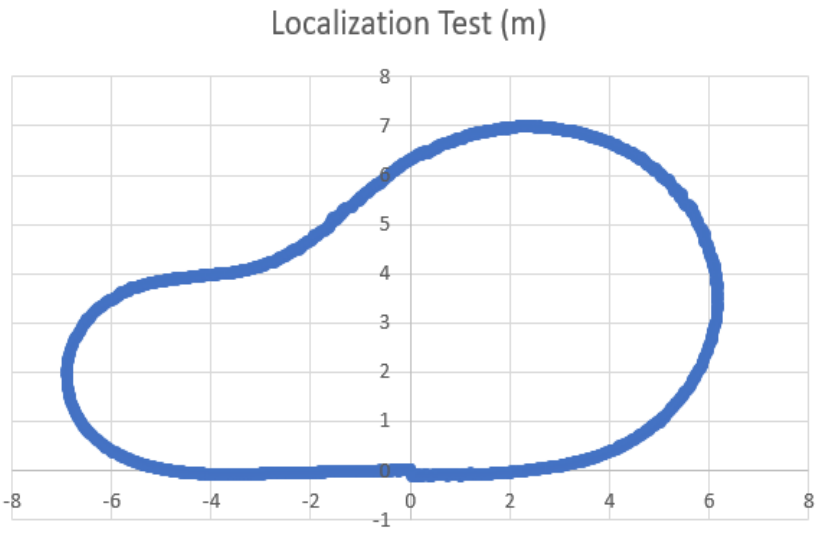

Figure 11 Visualize Test result: LiDAR on a Dolly

3.3 Map Test

By visualizing the LiDAR- Hector SLAM output of the map, and comparing to the real-world condition, we can verify the map generation method of our algorithm.

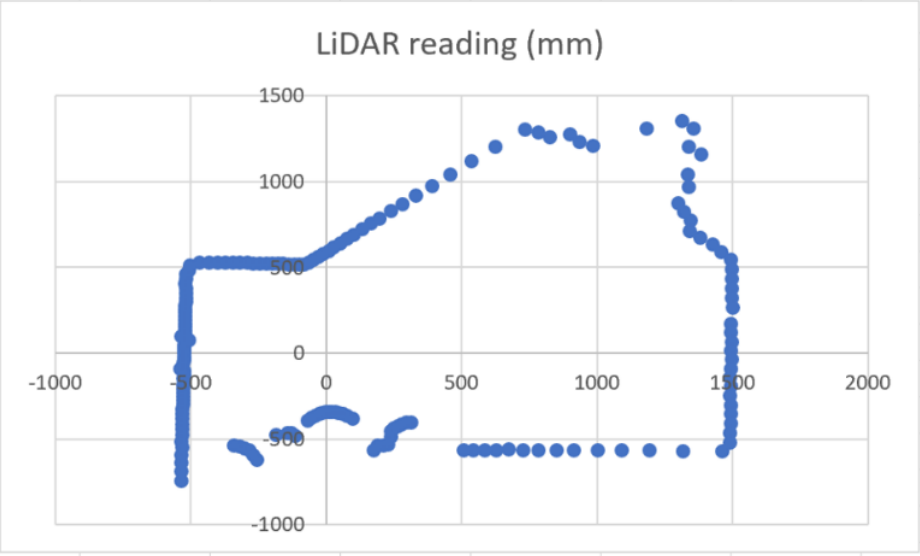

Figure 12 Point cloud of He's bathroom

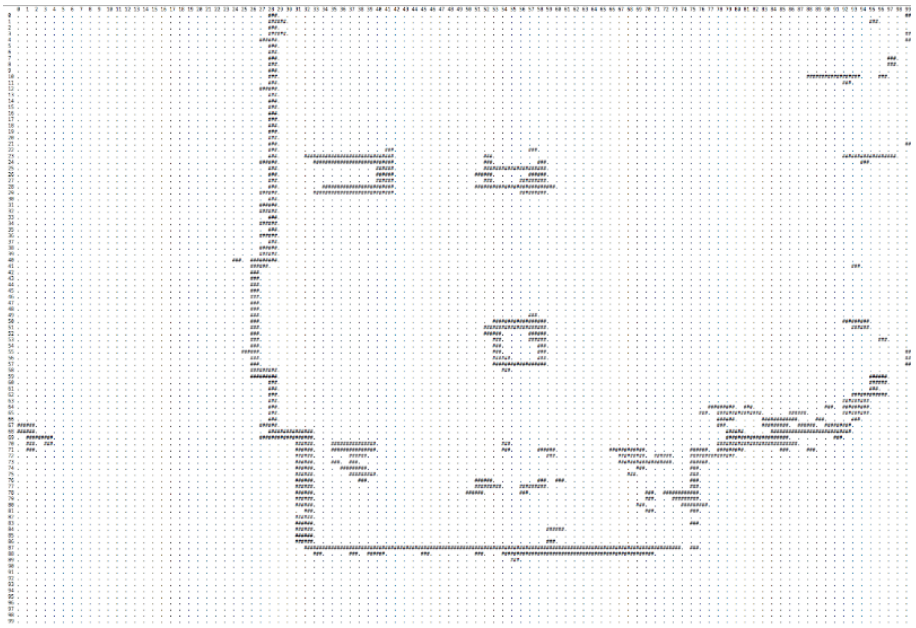

Figure 13 Map of the demo room

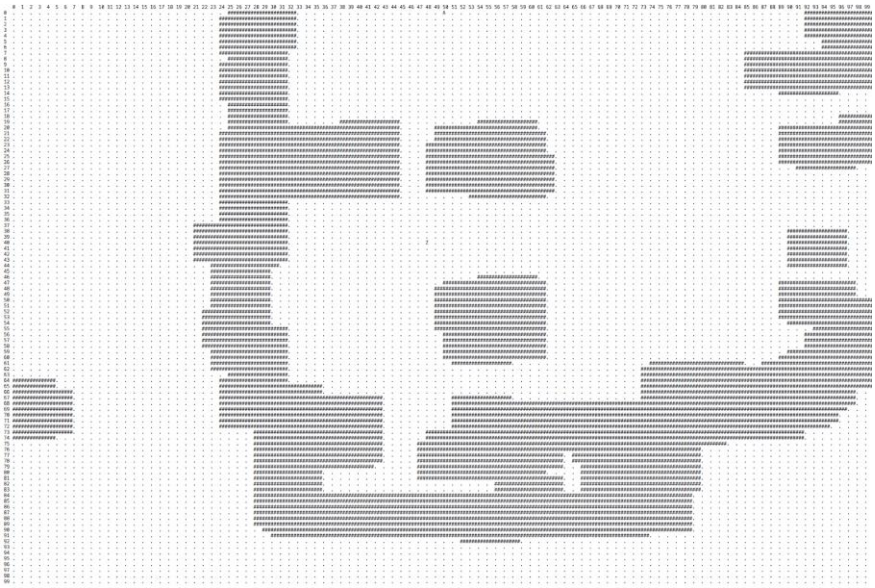

Figure 14 Test of object population

3.4 Path Planning Test

By comparing the visualization tool, we can identify if this path is an ideal path and we can also check if our path compression works correctly.

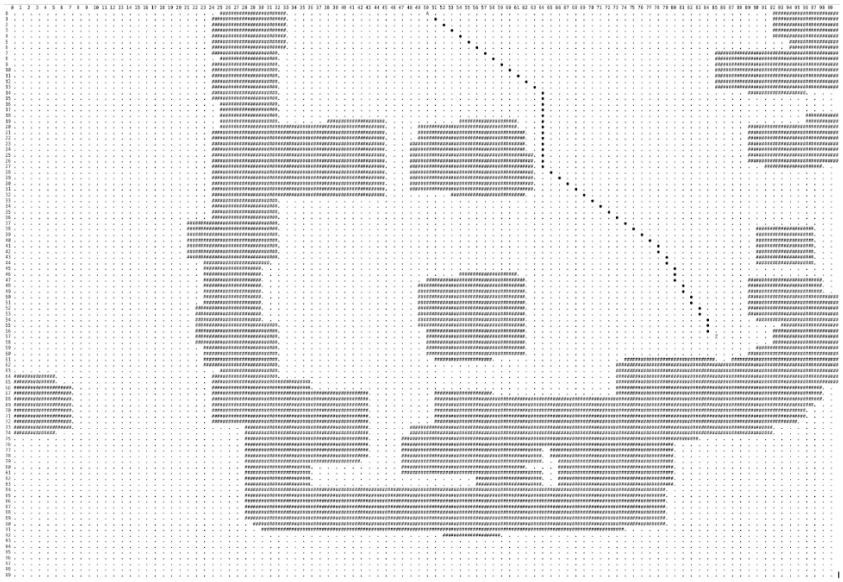

Figure 15 Sample path in a given map

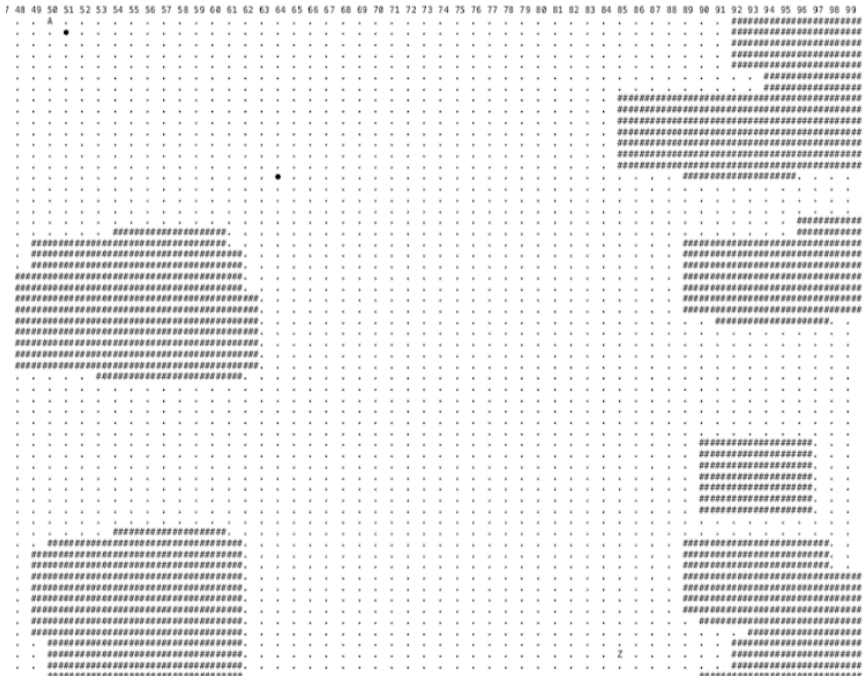

Figure 16 Returned path after compression

3.5 Position Control Test

In order to test the accuracy of the position control, we manually input a destination at (1,1) to the position control node. In the lab, the destination is marked on ground, and we measured the distance between the center of the LiDAR and the marked position. By ten testing trials, we get the following results:

Table 1 Position Test Result

Distance (cm)	1.6	1.1	1.0	2.0	2.1	1.5	1.8	1.5	2.0	1.8

The average distance is 1.64cm, which is accurate enough for the upper part to perform the grabbing process.

4 Costs

4.1 Parts

Table 2 Parts Costs

Part	Manufacturer	Number	Retail Cost ($)	Bulk Purchase Cost ($)	Actual Cost ($)
LiDAR	SLAMTEC	1	319.95	319.95	319.95
Chassis & Motors	Chihai	1	117	117	117
5v Battery	ANKER	1	39	39	39
Raspberry Pi	Raspberry Pi Foundation	5	54.99	54.99	274.95
12v Battery	Gissaral	1	69.95	69.95	69.95
Total					

4.2 Labor

Table 3 Labor Information

LABOR			
Name	**Salary ($/hour)**	**Hours**	**Total(*2.5) ($)**
Honglu He	40.00	130	13000
Chengliang Li	40.00	130	13000
Siping Meng	40.00	130	13000
TOTAL COST			39000

5 Conclusion

5.1 Accomplishments

We have achieved our basic requirements for individual module, but there are still further improvements needed to make this product adapted to different environment. Overall the project was successful and can complete a task without human interference.

Figure 17 Robot Vehicle

5.2 Uncertainties

At the very beginning of the whole process when the LiDAR scans for the first version of the map, if there are objects moving fast in a distance less than 10cm to the LiDAR, the original map that the robot generated will be messed up. Further inspection and experiment are needed.

5.3 Ethical considerations

We have several safety concerns about our project. The Li-po battery requires the highest attention to deal with due to the possibility of explosion, and we have made sure the temperature of the battery stay in the safe range of the industrial standard for all time. Our charger is an industry made IC device and shut charge controller off if charging input is beyond required voltage range. In this way, it can reduce the likelihood of hazards while charging. We will also design a protect circuit for sensor and motor components. To protect the circuit and PCB, we used fly-back diode and Transient Voltage Suppressor since the motor will also be connected to the PCB. What's more, in order to prevent a short circuit which may lead to electric shock, we

followed the electricity using manual during the experiment, and also checked our power and circuit before connecting to the battery every time before operation. .

Another safety concern is about the vehicle. Since we used four motors (2 for wheels and 2 for robot arm), we have to put malfunction of these parts into serious thoughts. If any software or circuit drive these motors mistakenly, the whole robot will be out of control and may hurt people standing nearby. This is what we need to avoid during the whole designing and demo period. In order to prevent such a condition, we first accomplished a circuit test for each part separately. Then we tested the searching algorithm and robot control code in a safe laboratory as many times as possible.

For the ethical issues, we followed IEEE and ACM code of ethics. We encountered many problems in the project. But when problems occurred, we revised and improved our design to fix the problem instead of disguising the problems and moving on recklessly. Based on #5 of the IEEE Code of Ethics, “to seek, accept, and offer honest criticism of technical work, to acknowledge and correct errors” [4]. Therefore, when problems showed up, we tried our best to find a way with teammates to solve them. If we could not solve the problem by ourselves, we turned to our TA for help.

5.4 Future work

Concise wiring and exposed components
Since the robot is used for the regular living room, and all the components and wires are exposed, we need to design an outer shell for protection.

Better control for motors
May need to implement DP control for the motor speed and veering.

More precision in localization
There is still 0.04m error for our SLAM algorithm, we can revise our algorithm for better accuracy.

Emergency stop
We can also design a stop function for sudden appear moving obstacles, to protect our robot and for better performance for more conditions.

6 References

[1] *E. Guizzo, "So, Where Are My Robot Servants?", May, 2012. [Online]. Available: https://spectrum.ieee.org/robotics/home-robots/so-where-are-my-robot-servants. [Accessed Dec. 2018]*.

[2] *S. Kohlbrecher, O. Von Stryk, J. Meyer, and U. Klingauf, "A flexible and scalable SLAM system with full 3D motion estimation," in Safety, Security, and Rescue Robotics (SSRR), 2011 IEEE International Symposium on, Kyoto, Japan, 2011, pp. 155-160.*

[3] *En.wikipedia.org. (2018). Conversion between quaternions and Euler angles. [online] Available at: https://en.wikipedia.org/wiki/Conversion_between_quaternions_and_Euler_angles [Accessed 1 Dec. 2018]*.

[4] *Ieee.org. (2018). IEEE IEEE Code of Ethics. [online] Available at: https://www.ieee.org/about/corporate/governance/p7-8.html [Accessed Dec. 2018]*.

7 Appendix A Requirement and Verification Table

Table 4 System Requirements and Verifications

Requirement	Verification	Verification status (Y or N)
1. The motor controller can receive the signal from Raspberry Pi and control the voltage and current direction across the motors. 2. All the parts on PCB can work	1. We can use the microcontroller to send the signal to the motor controller and control the motor rotation. 2. We can compare the parameters we set and the result we get when measuring	Y
1. The motor can drive the robot with all the components on it and the speed should be no less than 4.7cm/s i.e. robot can move along the diagonal line in 2mins.	1. We test the time of the robot with all the components on crossing the diagonal line and check if the time is less than 2mins	Y
1. The robot will not hit obstacles during the whole process, including walls. 2. The robot will walk down the optimal path.	1. By observation, the robot should not hit any objects. 2. Robot will send back its current path to show this on 2d map.	Y
1. The whole power set need to provide steady and plentiful power. The power converter, which is the DC-DC converter, need to	1. When we test the voltage out from the DC-DC converter, we should get the voltage with the difference between the expectation less than 5%	Y
1. The robot should mark any obstacle it can see from its prospective	A. Putting the robot at home position for 5 seconds, then check the updated map, and compare it with actual environment B. Start the robot at home position and give an object location, check if the map is updated during the run	Y
1. The robot location should be +/-0.05m from actual location	1. We set a object location and let the robot drive itself, check its final location and the given object location	Y